Performance testing

chengbaolei@suda.edu.cn

Review

- In Session 12-13:
  - Unit testing
  - Integration testing
  - System testing
  - Acceptance testing
  - Regression testing

Summary

- 各用一句话分别描述UISA的基本定义
- 结合5W+1H对比UISA
- 理解回归测试，分别论述TC-Minimization, TC-Selection, TC-Prioritization的特点和做法
- 对比Smoke Testing 和 Sanity Testing

Summary

1. 软件测试主要由单元测试、集成测试、系统测试、验收测试等过程组成，它与软件开发的周期相关
2. 单元测试是测试的第一项工作，重点测试模块的正确性，它是软件能正确运行的基础

   桩模板、驱动模块
3. 集成测试是按某种策略，将通过单元测试的模块集成起来所进行的正确性测试，其重点是测试模块间的接口

   各种集成策略

Summary

4. 系统测试是将经过集成测试后的软件与其运行环境、支撑环境及人组成一个完整系统的测试，重点测试软件的功能、性能、行为等方面　性能测试

5. 验收测试更多的是从用户的角度对经过系统测试的软件在运行了一段时间后再对其功能、性能等方面的测试，考察其是否满足原先的需求　α/β测试

6. 从单元测试到验收测试，主要测试任务、参与人员、测试技术、测试数据等方面的变化

7. 回归测试是一项面广量大的测试工作，其测试效率非常重要　TC-M, TC-S, TC-P

内容提要

- 理解“性能”与“性能测试”
- 压力测试
- 性能测试的过程与实践

功能测试 vs 性能测试

	功能测试	性能测试
测试目的	1.验证系统是否满足需求 2.查找系统中可能存在的功能缺陷	1.验证系统是否满足需求 2.查找系统中可能存在的性能缺陷或瓶颈
测试依据	系统的功能需求	系统的性能需求
测试过程	分析系统需求->设计测试用例模拟用户场景->执行测试->提交缺陷->验证缺陷是否解决	分析系统需求->设计性能测试用例模拟用户场景->执行测试->提交缺陷->验证缺陷是否解决
行业背景	要求对业务和用户场景非常熟悉	要求对业务和用户场景非常熟悉

什么是性能测试？

- 性能测试（performance test）就是为了获取系统性能相关指标或发现系统性能问题而进行的测试
- 一般在真实环境、特定负载条件下，通过工具模拟实际软件系统的运行及其操作，同时监控性能各项指标，最后对测试结果进行分析来确定系统的性能状况
- 观察系统在一个给定的环境和场景中的性能表现是否与预期目标一致，评判系统是否存在性能缺陷，并根据测试结果识别性能瓶颈，改善系统性能的完整的过程

性能测试目标

- 获取系统性能某些指标数据
- 为了验证系统是否达到用户提出的性能指标
- 发现系统中存在的性能瓶颈，优化系统的性能

与教学过程相似

性能测试类型

- 性能验证测试，验证系统是否达到事先已定义的系统性能指标、能否满足系统的性能需求
- 性能基准测试，在系统标准配置下获得有关的性能指标数据，作为将来性能改进的基准线
- 性能规划测试，在多种特定的环境下，获得不同配置的系统的性能指标，从而决定在系统部署时采用什么样的软、硬件配置

一些常见的性能问题

- 症状
  - 启动系统、打开页面越来越慢
  - 查询数据，很长时间才显示列表
  - 网络下载速度很低，如5k/s
  - …
- 原因
  - 资源耗尽，如CPU使用率达到100%
  - 资源泄漏，如内存泄漏，最终会导致资源耗尽
  - 资源瓶颈，如线程、GDI、DB连接等资源变得稀缺

如何评价系统的性能

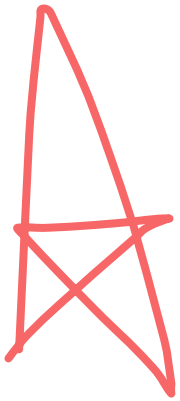
- 用户(end-user)的视角
  - 响应时间(Response Time)
- 运营商(customer)和开发商的视角
  - 响应时间(Response Time)
  - 并发用户数(The Number of Concurrent Users)
  - 吞吐量(Throughput) – 每秒交易数(Transaction per Second)
  - 资源利用率(Hardware/Software Resource Utilization)
  - 可靠性或稳定性(Reliability or Stability)
  - 可伸缩性(Scalability)
  - 可恢复性(Recoverability)

可伸缩性是指系统通过增加或减少硬件水平从而提升或降低系统性能的难易程度。可伸缩性分为scale up和scale out。
scale up是指提高单台服务器的硬件水平来提高系统的整体处理能力，可以调整的有CPU，存储，内存等；scale out是指通过增加系统的处理节点的方式来提高系统的整体处理能力。

内容提要

- 理解“性能”与“性能测试”
- 压力测试
- 性能测试的过程与实践

压力/负载测试



- 压力测试(Stress test)，也称为强度测试、负载测试。
- 压力测试是模拟实际应用的软硬件环境及用户使用过程的系统负荷，长时间或超大负荷地运行测试软件，来测试被测系统的性能、可靠性、稳定性等。
- 在一种需要反常（如长时间的峰值）数量、频率或资源的方式下，执行可重复的负载测试，以检查程序对异常情况的抵抗能力，找出性能瓶颈或其它不稳定性问题。

压力测试类型

- 并发性能测试（重点）
- 疲劳强度测试
- 大数据量测试

并发性能测试

- 并发性能测试的过程也是一个负载测试过程，即逐渐增加并发虚拟用户数负载，直到系统出现性能瓶颈或者崩溃为止。
- 破坏性压力测试，通过不断加载的手段，快速造成系统的崩溃，让问题尽快地暴露出来。
- ramp-up测试 设置在多长时间内建立全部的线程。假设ramp-up period 设置成T 秒， 全部线程数设置成N个，将每隔T/N秒建立一个线程。

疲劳强度测试

- 采用系统稳定运行情况下能够支持的最大负载，持续长时间运行，以发现性能问题。
- 渗入测试（soak test），通过长时间运行，使问题逐渐渗透出来，从而发现内存泄漏、垃圾收集（GC）或系统的其他问题，以检验系统的健壮性
- 峰谷测试（peak-rest test），采用高低突变加载方式进行，先加载到高水平的负载，然后急剧降低负载，稍微平息一段时间，再加载到高水平的负载，重复这样过程，容易发现问题的蛛丝马迹，最终找到问题的根源

大数据量测试

- **独立的数据量测试**
  - 针对某些系统存储、传输、统计、查询等业务进行大数据量测试

- **综合数据量测试**
  - 和压力性能测试、负载性能测试、并发性能测试、疲劳性能测试相结合的综合测试方案

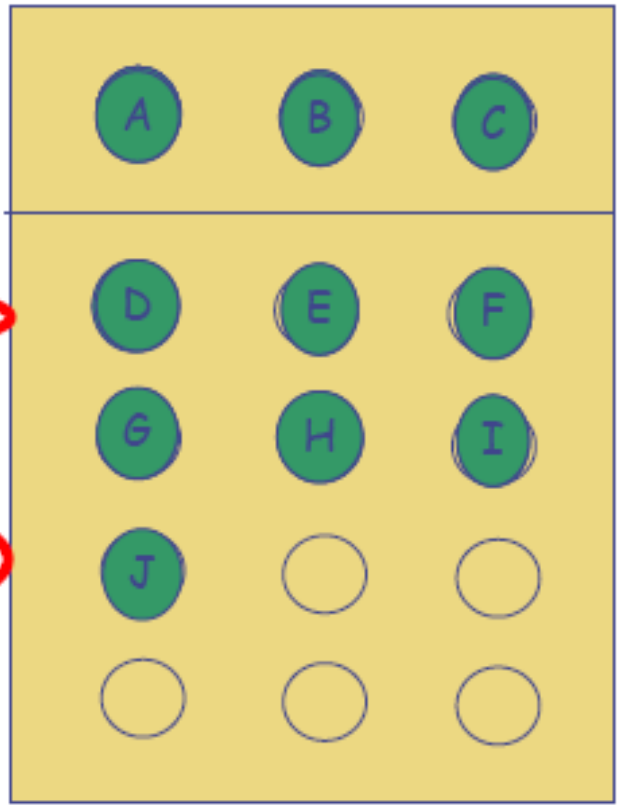
- 大数据量测试：10万、100万、千万条记录
- 大容量测试：某些字段存储10M、100M、1G等大体积数据

理发店模型的3个假设

- 理发店中一共有 3 名理发师
- 每位理发师剪一个发的时间都是 1 小时
- 我们顾客们都是很有时间观念的人而且非常挑剔，他们对于每次光顾理发店时所能容忍的等待时间+剪发时间是3小时，而且等待时间越长，顾客的满意度越低。如果3个小时还不能剪完头发，我们的顾客会立马生气的走人

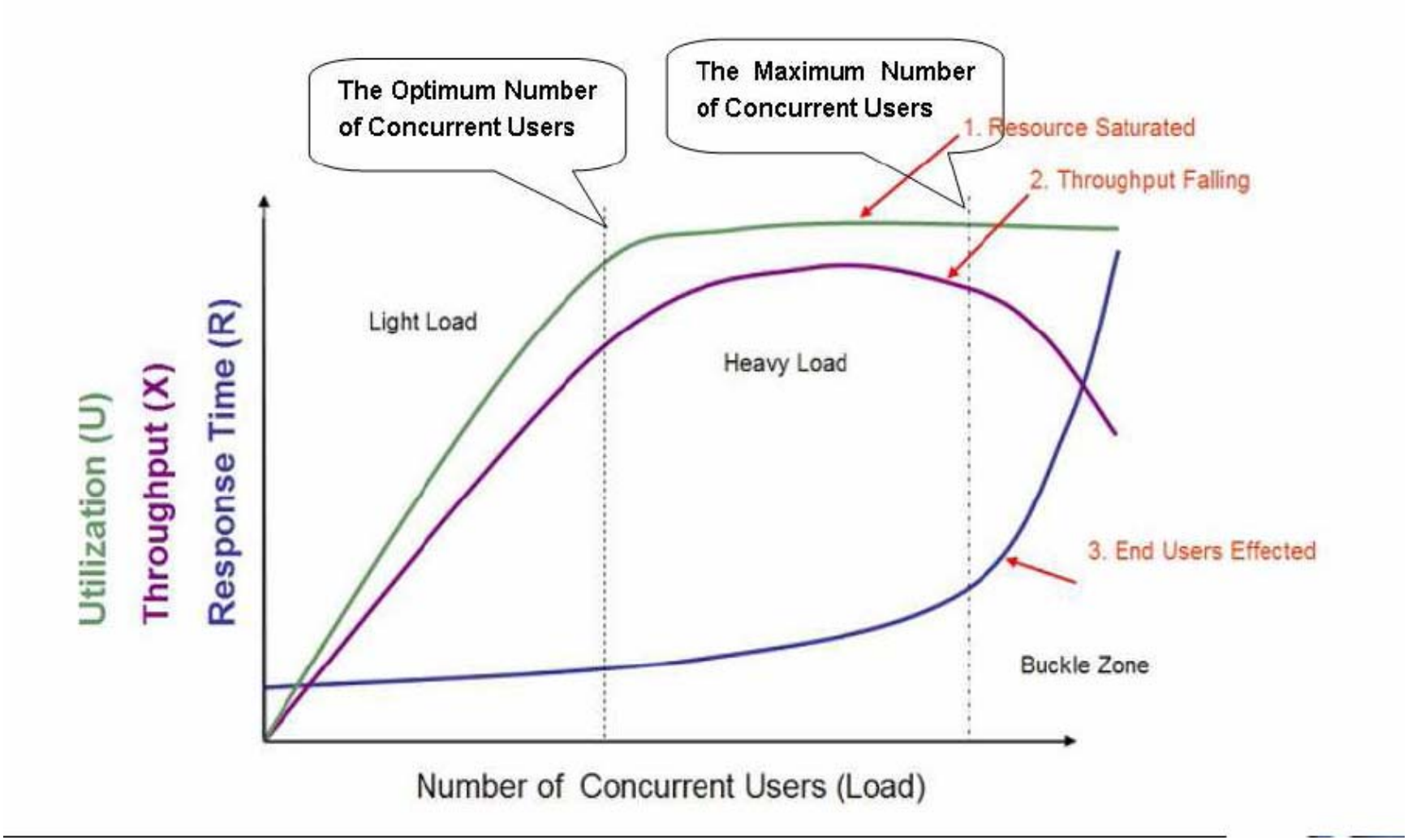
理发店内场景的模拟

顾客数	理发师状态	顾客状态
1	有两名理发师空闲	顾客不需要等待
2	有一名理发师空闲	顾客不需要等待
3	没有理发师空闲	没有顾客需要等待
4	没有理发师空闲	有1名顾客需要等待
9	没有理发师空闲	接近顾客所能忍受的最大限度，顾客满意度接近最低
10	没有理发师空闲	必然有一位顾客最终会选择离开



如果持续 每个小时>3个 顾客？

理发店的性能模型


The Optimum Number of Concurrent Users
The Maximum Number of Concurrent Users
1. Resource Saturated
2. Throughput Falling
Light Load
Heavy Load
3. End Users Effected
Buckle Zone
Utilization (U)
Throughput (X)
Response Time (R)
Number of Concurrent Users (Load)

最佳/最大并发用户数

- 最佳并发用户数
  - 应当大于系统的平均负载
  - 当并发用户数持续大于该值，可能会出现部分用户请求失败
- 最大并发用户数
  - 应当大于系统的峰值负载
  - 当并发用户数大于该值，则必然会有用户请求失败

负载模型

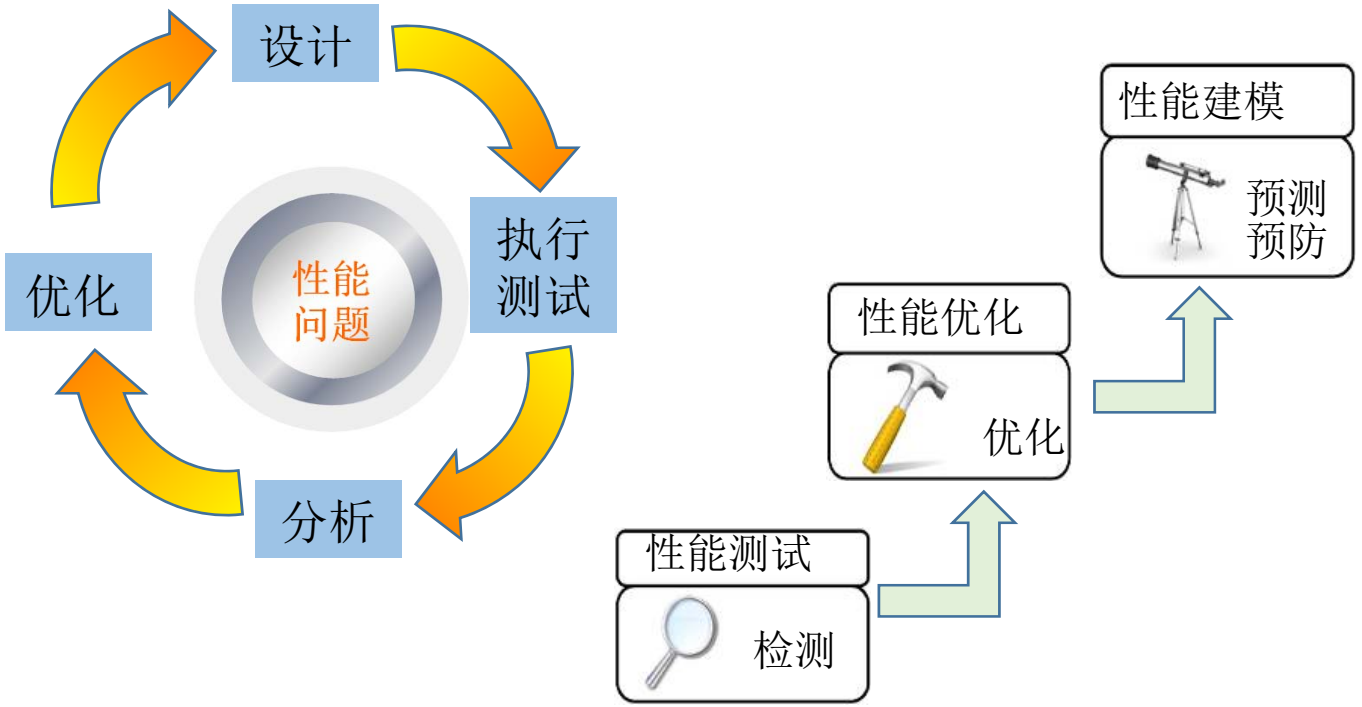
- 并发用户数量+思考时间+每次请求发送的数据量+负载模式

- 并发用户数量：近似于在线用户数量—Vusers
- 思考时间：请求间隔时间
- 负载模式：一次加载，递增加载，高低突变加载，随机加载

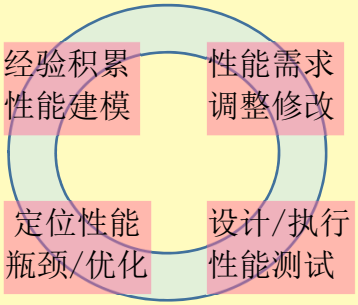
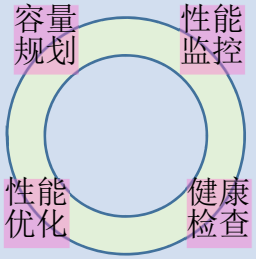
内容提要

- 理解“性能”与“性能测试”
- 压力测试
- 性能测试的过程与实践

持续性能测试

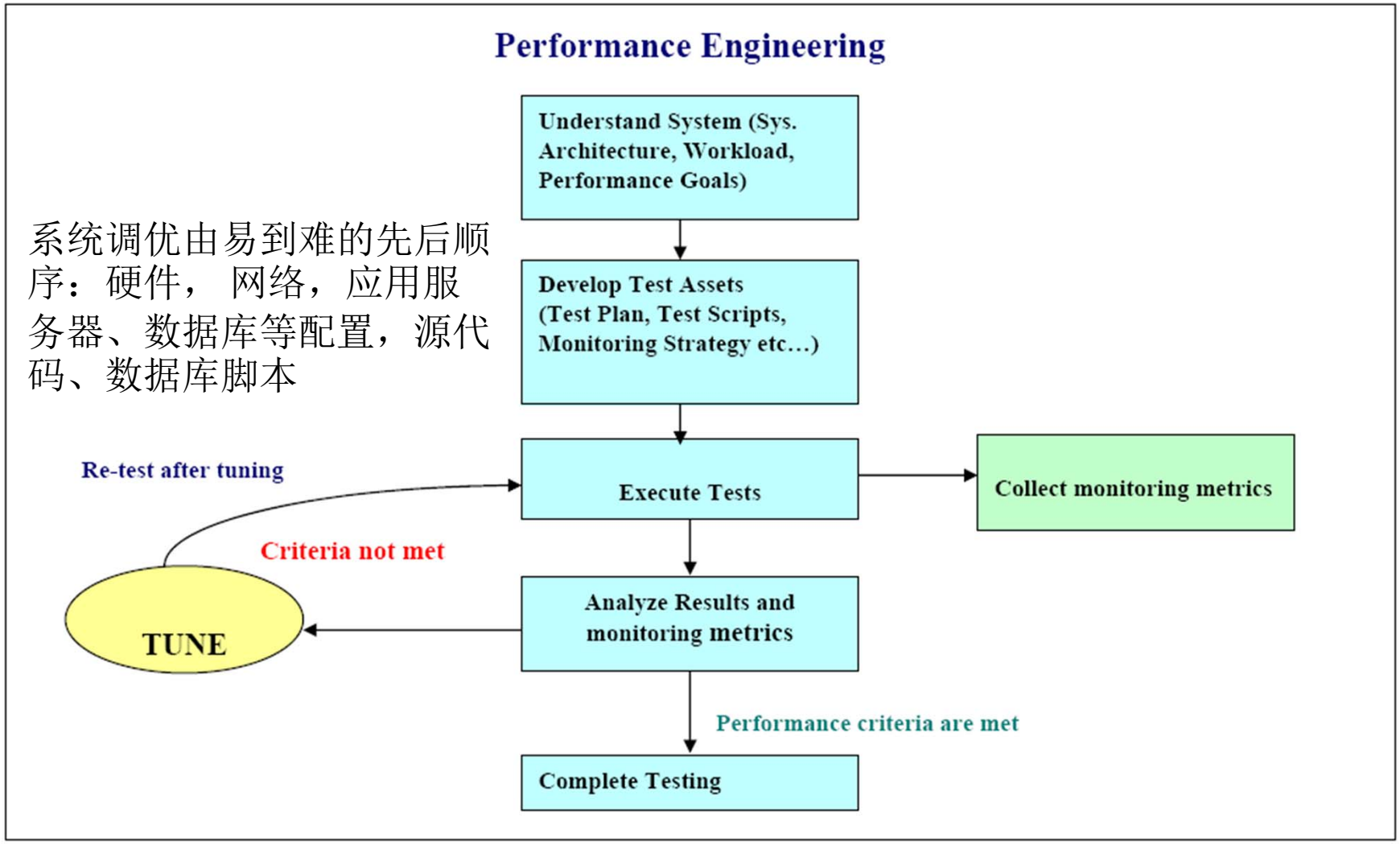


全生命周期性能评估

计划	系统架构	迭代	在线
确定性能需求 （包括运维需求） 业务场景 业务数据 性能测试策略 性能需求和计划评审	架构评估 性能测试策略	 性能数据	 性能数据

Performance Engineering Methodology

系统调优由易到难的先后顺序：硬件， 网络，应用服务器、数据库等配置，源代码、数据库脚本



性能测试需求和指标

- **性能测试需求：**

  用户对各项指标提出的明确需求；如果用户没有提出性能指标，则根据用户需求、测试设计人员的经验来设计各项测试指标。（需求+经验）

- **主要的性能指标：**

  服务器的各项指标（CPU、内存占用率等）、后台数据库的各项指标、网络流量、响应时间……

  参考P152

确定性能需求

- 只有具备了清楚而量化的性能指标，性能测试才能开始实施。

- 最终用户的体验，如2-5-10s原则
- 商业需求，如“比竞争对手的产品好”
- 技术需求，如CPU使用率不超过70%
- 标准要求

- 响应时间是用户的关注点
- 容量和数据吞吐量是（产品市场团队）业务处理方面的关注点
- 系统资源占用率是开发团队的技术关注点

性能需求应具有可测试性

每天支持1亿交易量，所有交易响应时间在1秒钟以内，支持1.5亿用户。

每天支持1亿交易量，正常业务模型最高支持1000TPS (transaction per second)；

本地交易高峰响应时间在1秒钟以内，跨行交易响应时间在2秒钟以内；

支持1.5亿用户，支持100万在线用户；

交易成功率99.99%以上；

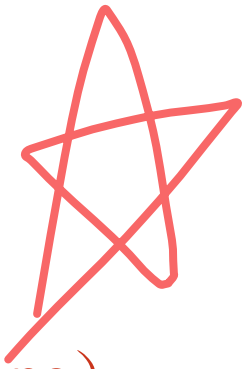
资源利用率：CPU 70%以内、内存80%以内……

上下文驱动：不同业务场景指标，不同业务支持指标

如何获得可测试的性能需求

- 客户方提出
- 根据历史数据来分析
- 参考历史项目的数据
- 参考其他同行类似项目的数据
- 参考其他类似行业应用的数据
- 参考新闻或其他资料中的数据
- **80/20** 原则

性能的具体指标



- 数据传输的吞吐量（Transactions）
- 数据处理效率（Transactions per second）
- 数据请求的响应时间（Response time）
- 内存和CPU使用率
- 连接时间（Connect Time）、发送时间（Sent Time）
- 处理时间（Process Time）、页面下载时间
- 第一次缓冲时间
- 每秒（SSL）连接数
- 每秒事务总数、每秒下载页面数
- 每秒点击次数、每秒HTTP 响应数
- 每秒重试次数

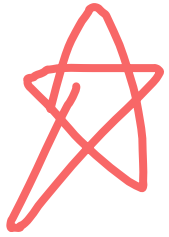
吞吐量

- 一次性能测试过程中网络上传输的数据量的总和
  - 如一个大型工厂，一天生产10W吨的货物，一天拉2吨的货物。运输能力是整个系统的瓶颈。
  - 一个水龙头开一天一夜，流出10吨水；10个水龙头开1秒钟，流出0.1吨水。一个水龙头的吞吐量大。
- 用吞吐量来衡量一个系统的输出能力是极其不准确的，因此加单位时间--QPS/TPS

QPS/TPS

- 每秒钟request/事务
- 完成请求的数量/完成请求所花费的时间
- 如果10秒的时间内，系统接受到了3000个请求，返回了2000个，剩下1000报错。则：

  QPS=2000/10=200



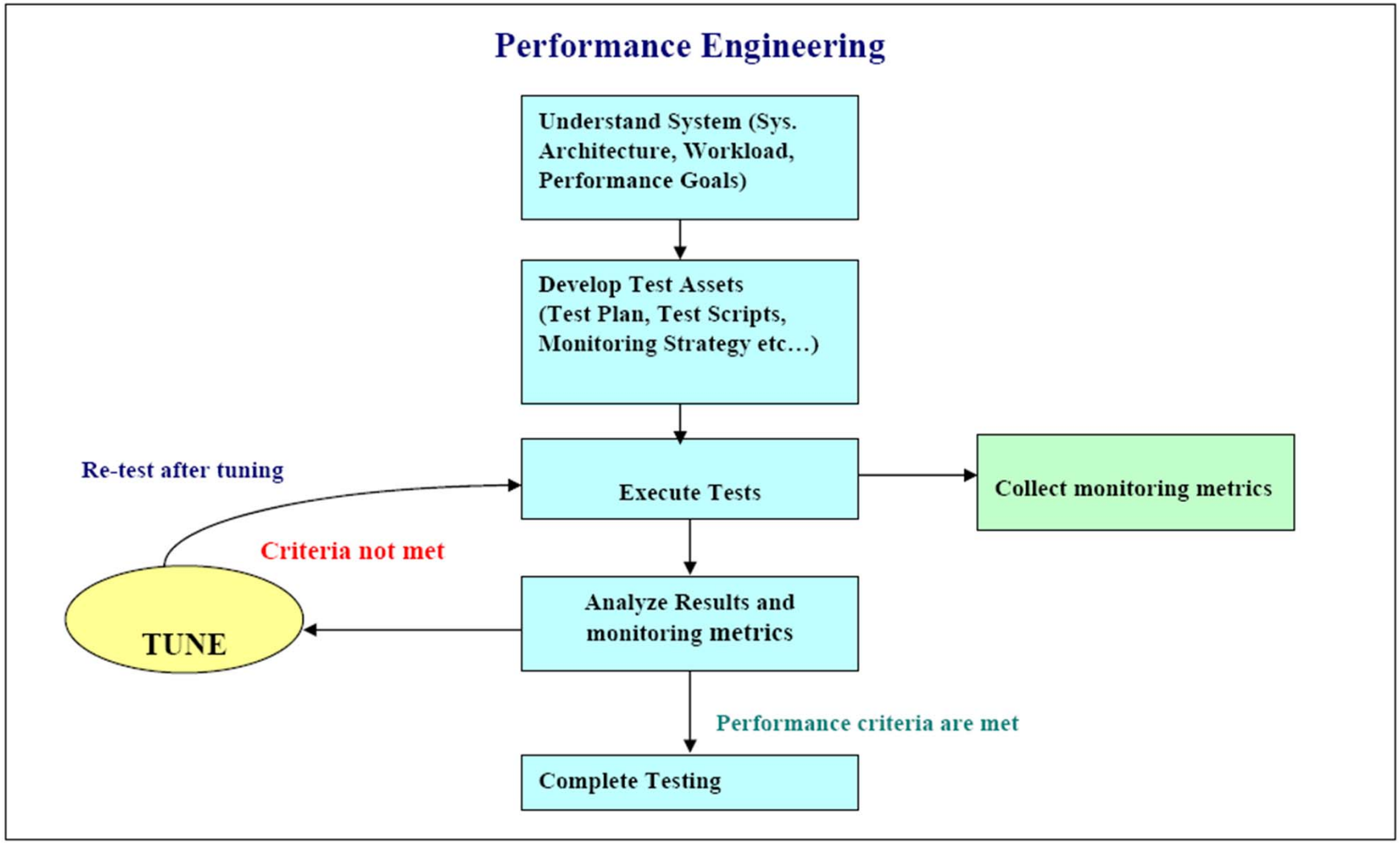
响应时间

- 单个请求:服务响应一次请求的花费的时间。考虑完成所有请求的平均响应时间及中位数时间。
- 平均响应时间:完成请求花费的总时间/完成的请求总数。
  - 响应时间分别是 {1，3，5，10，16} 和 {5，6，7，8，9}，
- 中位数响应时间:意味着至少有50%的数据低于或高于这个中位数。
- TP – Top Percentile
  - TP50: 50%的请求都小于某个值
  - TP90: 90%的请求都小于某个值

并发数

- 在某一时间点，服务器正在处理的请求数
  - 工程师经常说1秒并发2000: QPS=2000
  - 网站管理员说我们并发1000人，其实指的最大在线人数1000人。
  - 运维人员说我设置的tomcat的并发数500。但是受限于CPU、OS等其他原因，并发数在实际中达不到这个数值
  - 性能测试人员说我在LR中设置了并发数3000。但是客户端受限于CPU、OS等其他原因，并不能达到此压力。从服务器的角度出发，会有排队机制以及部分请求异常溢出，并不能说服务器的并发就到了3000。
- 通常，性能测试中的并发数指的是一个测试出来的结果计算值，其计算公式是 QPS*平均响应时间

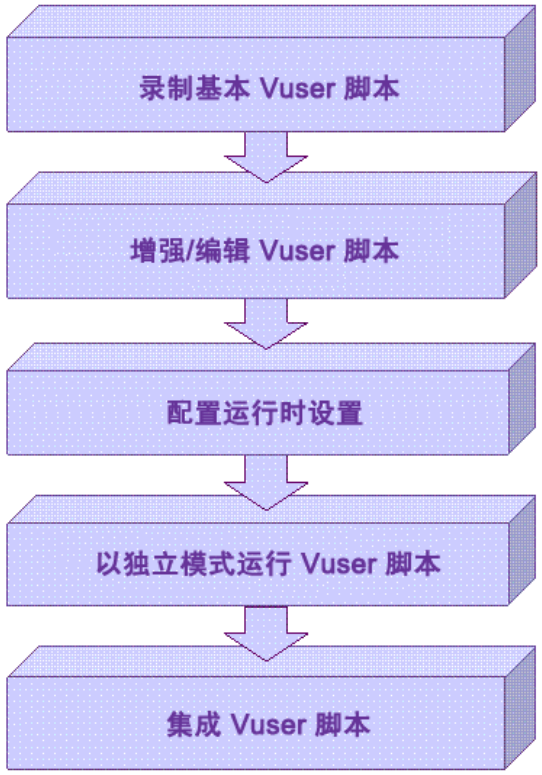
Performance Engineering Methodology



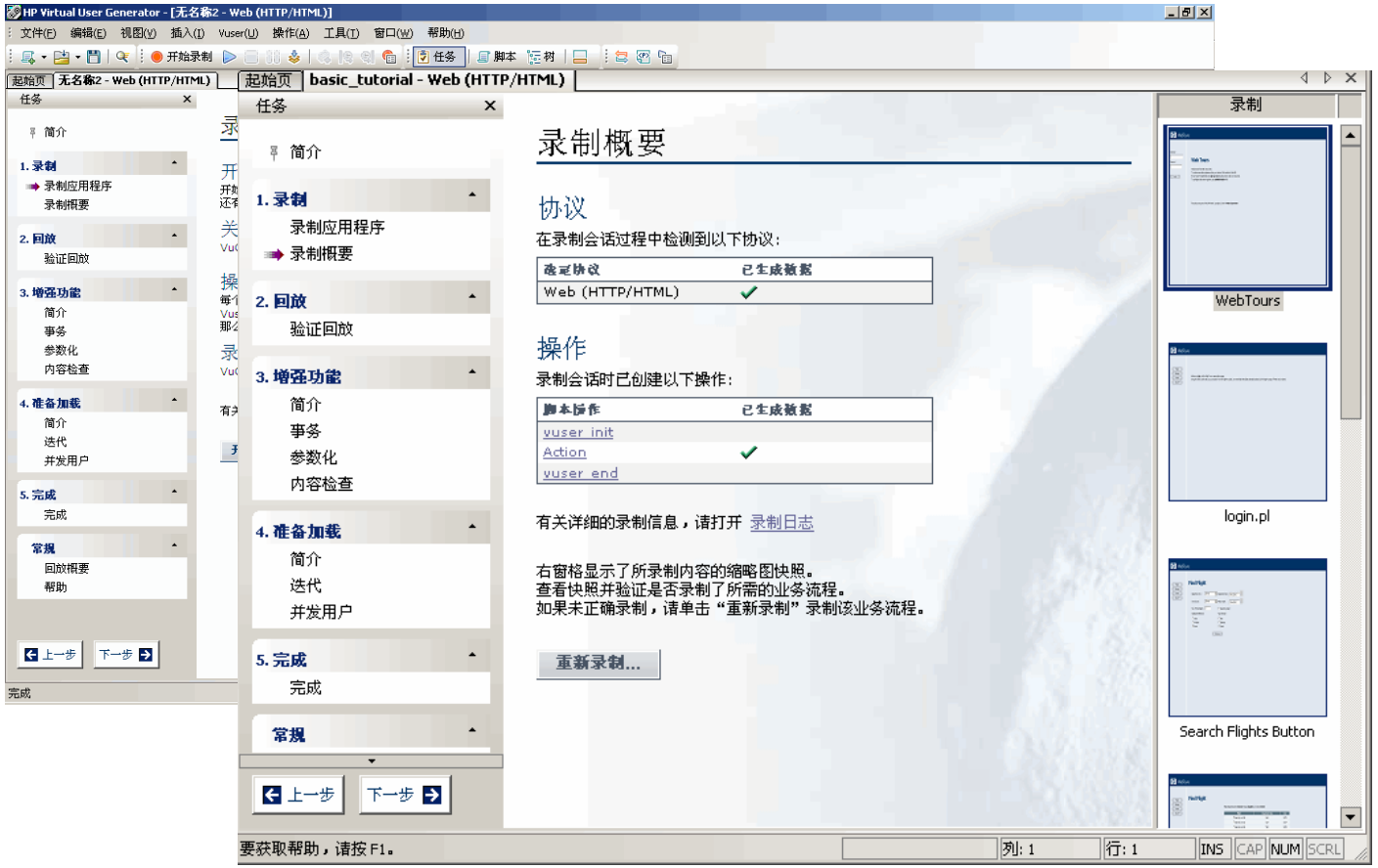
示例：性能测试过程


HP LoadRunner 11.00
LoadRunner
退出
配置
帮助
LoadRunner 启动程序
创建/编辑脚本
运行负载测试
分析测试结果
通过在部署系统或升级之前运行负载测试来检测瓶颈，可以避免实际运行时出现代价高昂且难以解决的性能问题。 通过在 QA 或其他阶段中的 IT 系统上创建工作负载，您可以评测关键业务流程在负载下的性能并找出需要快速解决的瓶颈。 HP 的 LoadRunner 软件全面支持各种常见协议和技术，包括 ERP/CRM、Web、J2EE/.NET、XML、.NET、无线和流媒体。通过 LoadRunner，允许 IT 团队使用一个产品对所有企业应用程序进行负载和性能测试。 使用 HP 的 LoadRunner 软件进行有效的负载测试可以帮助确保新的或升级的应用程序达到服务水平目标且能部署到实际应用环境而不出现性能问题。
了解有关负载测试的详细信息
新增功能
VuGen 中的新增功能
增强了对 Web 2.0 和 RIA 应用程序的支持:
新 Ajax TruClient 协议，适用于可在 Web 浏览器中模拟用户活动的基于 JavaScript 的现代 Web 应用程序。
新协议，适用于 Microsoft Silverlight 应用程序以及通过 HTTP 进行通信的 Java 应用程序。
Web (HTTP/HTML) 协议中增强了对数据格式的支持（例如，JSON）。
功能视频:
Ajax TruClient 协议简介
Web (HTTP/HTML) 协议支持高级数据格式
Silverlight 协议
Java over HTTP 协议
Controller 中的新增功能
增强的服务水平协议功能可支持事务响应时间百分比
Controller 现在允许自定义性能监控器的默认度量列表，以适应最优测试方法
功能视频:
服务水平协议百分比支持
自定义监控器默认计数器列表
Analysis 中的新增功能
增强的报告功能:
可完全自定义的报告模板
创建会话时自动生成报告的功能
功能视频:
新的报告功能
提高了性能和可用性

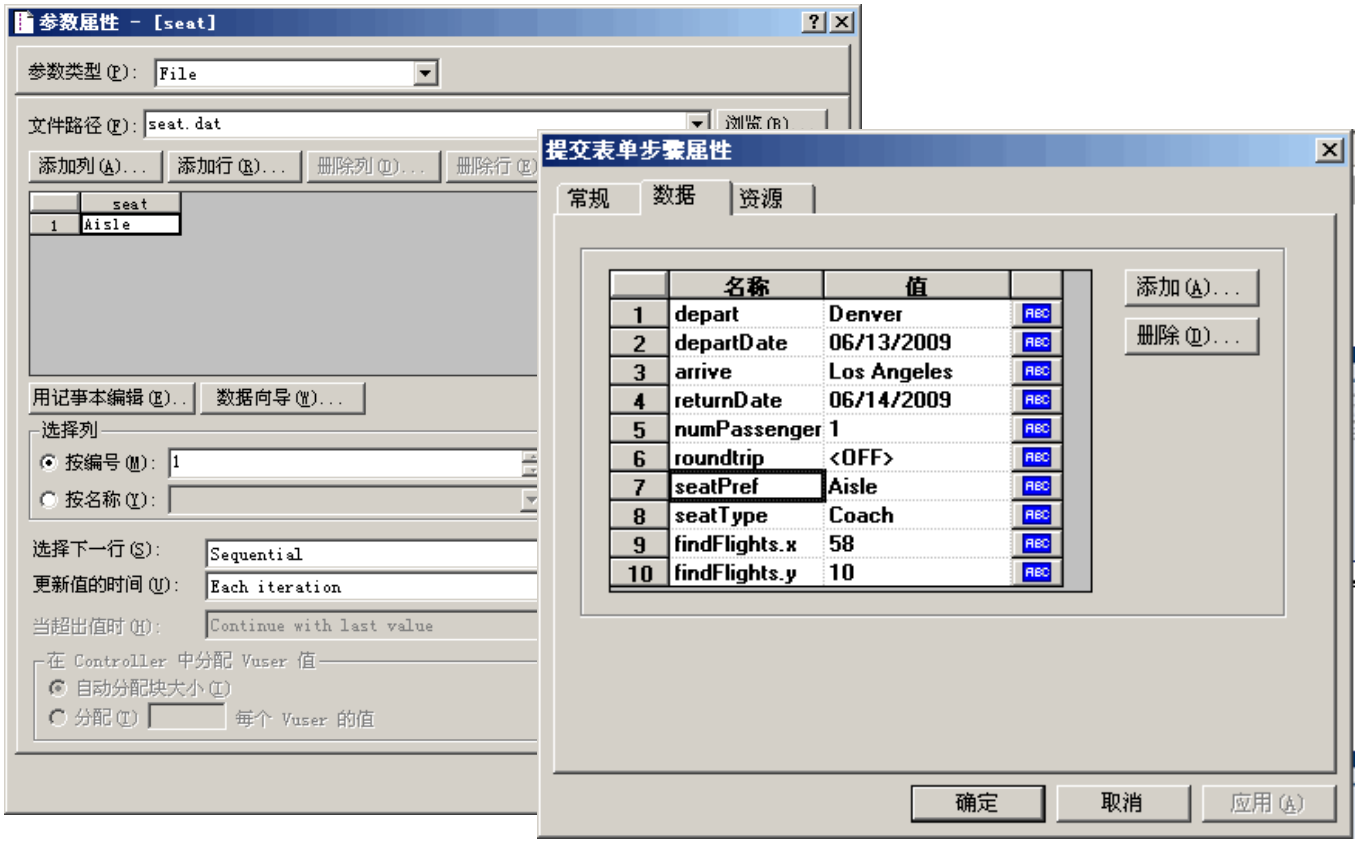
示例：脚本创建过程


录制基本 Vuser 脚本
增强/编辑 Vuser 脚本
配置运行时设置
以独立模式运行 Vuser 脚本
集成 Vuser 脚本

示例：录制脚本

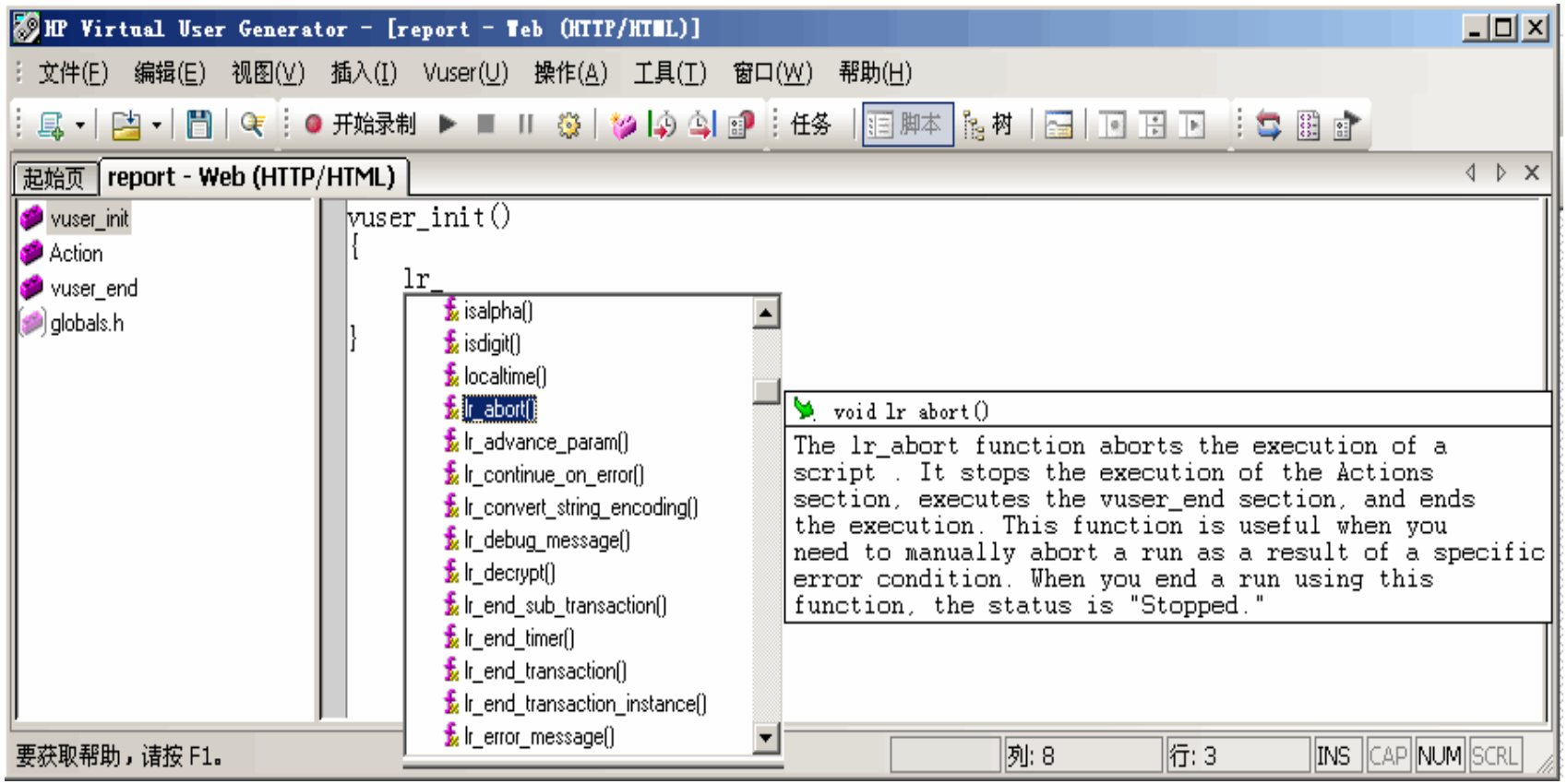

HP Virtual User Generator - [无名称2 - Web (HTTP/HTML)]
文件(F) 编辑(E) 视图(V) 插入(I) Vuser(U) 操作(A) 工具(T) 窗口(W) 帮助(H)
开始录制
任务
脚本
树
起始页
无名称2 - Web (HTTP/HTML)
basic_tutorial - Web (HTTP/HTML)
任务
简介
1. 录制
录制应用程序
录制概要
2. 回放
验证回放
3. 增强功能
简介
事务
参数化
内容检查
4. 准备加载
简介
迭代
并发用户
5. 完成
完成
常规
回放概要
帮助
上一步
下一步
录制概要
协议
在录制会话过程中检测到以下协议：
已生成脚本
Web (HTTP/HTML)
操作
录制会话时已创建以下操作：
已生成脚本
vuser_init
Action
vuser_end
有关详细的录制信息，请打开 录制日志
右窗格显示了所录制内容的缩略图快照。
查看快照并验证是否录制了所需的业务流程。
如果未正确录制，请单击“重新录制”录制该业务流程。
重新录制...
录制
WebTours
login.pl
Search Flights Button
要获取帮助，请按 F1。
列: 1
行: 1
INS
CAP
NUM
SCRL

示例：脚本参数化



创建参数列表，用参数替换固定的文本，模拟用户真实的业务操作。

示例：虚拟用户


HP Virtual User Generator - [report - Web (HTTP/HTML)]
文件(F) 编辑(E) 视图(V) 插入(I) Vuser(U) 操作(A) 工具(T) 窗口(W) 帮助(H)
开始录制
任务
脚本
树
起始页
report - Web (HTTP/HTML)
vuser_init
Action
vuser_end
globals.h
vuser_init()
{
lr_
}
isalpha()
isdigit()
localtime()
lr_abort()
lr_advance_param()
lr_continue_on_error()
lr_convert_string_encoding()
lr_debug_message()
lr_decrypt()
lr_end_sub_transaction()
lr_end_timer()
lr_end_transaction()
lr_end_transaction_instance()
lr_error_message()
void lr_abort()
The lr_abort function aborts the execution of a script . It stops the execution of the Actions section, executes the vuser_end section, and ends the execution. This function is useful when you need to manually abort a run as a result of a specific error condition. When you end a run using this function, the status is "Stopped."
要获取帮助，请按 F1。
列: 8
行: 3
INS
CAP
NUM
SCRL

What Do The Simulated Users Do?

Using Documentation, Interviews and Exploration…

Determine the System's Purpose and what Activities it is used for.

Then Measure or Estimate:

- All possible user activity.
- How often each activity is accomplished.
- All "types" of users.
- What activities are most performance intensive.
- Other user community modeling information.

So you can model "*Real*" users.

Why model real users?

What Do The Simulated Users Do?

Because:

Results from inaccurately modeled tests are nearly always inaccurate, and often lead to incorrect decisions.

The only way to predict actual user experience (end-to-end response time) is to execute tests using realistic User Community Model(s).

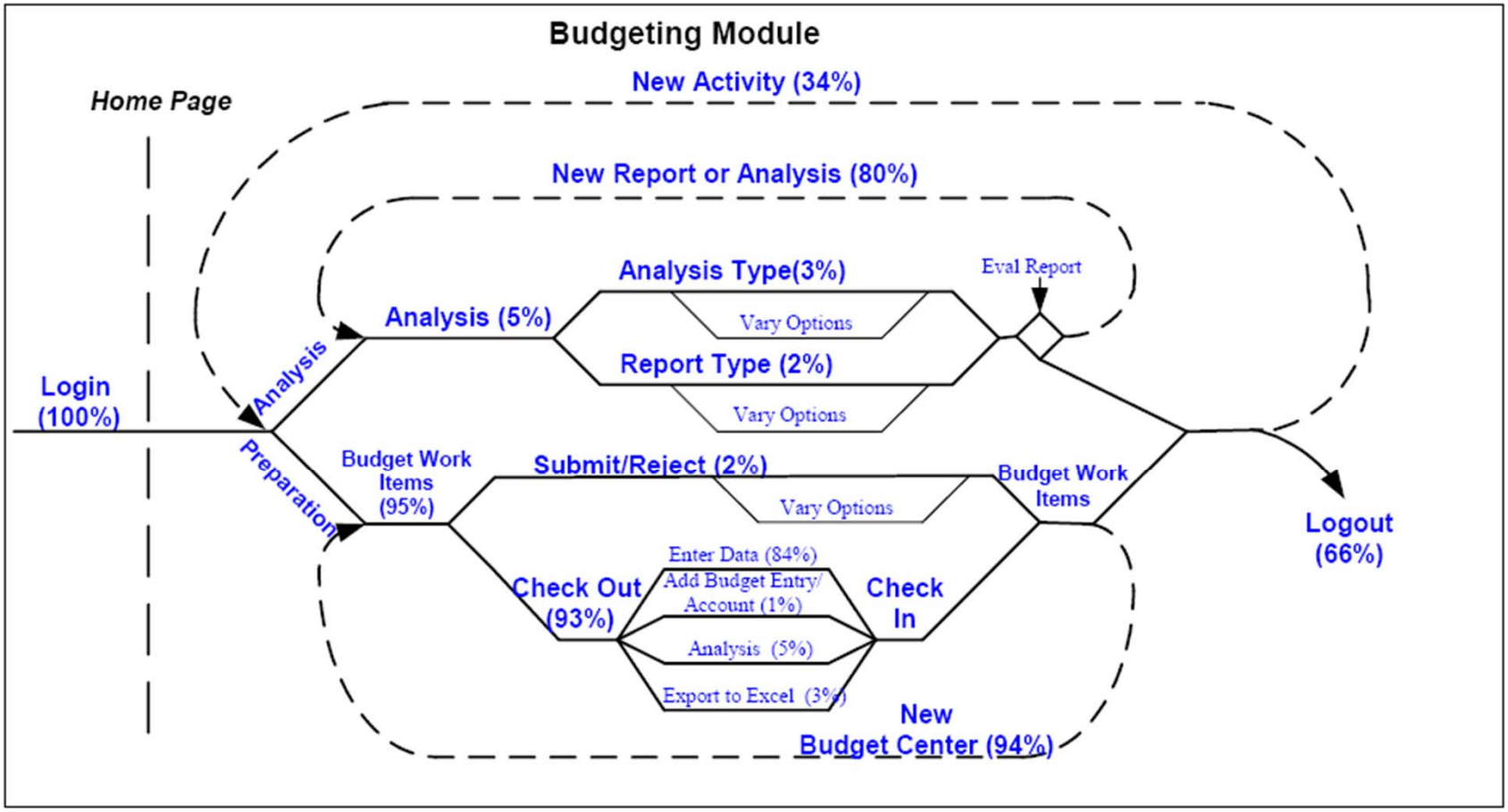
Extrapolating expected performance based on incomplete models doesn't work.

"The one thing that matters the most is not how your site behaves under theoretical or simulated conditions, but how well it works when you plug it into the wall and let everyone come hit your box from all across the world"

— Serdar Yegulalp in "Website Stress-Testing"

What Do The Simulated Users Do?

Build a visual model

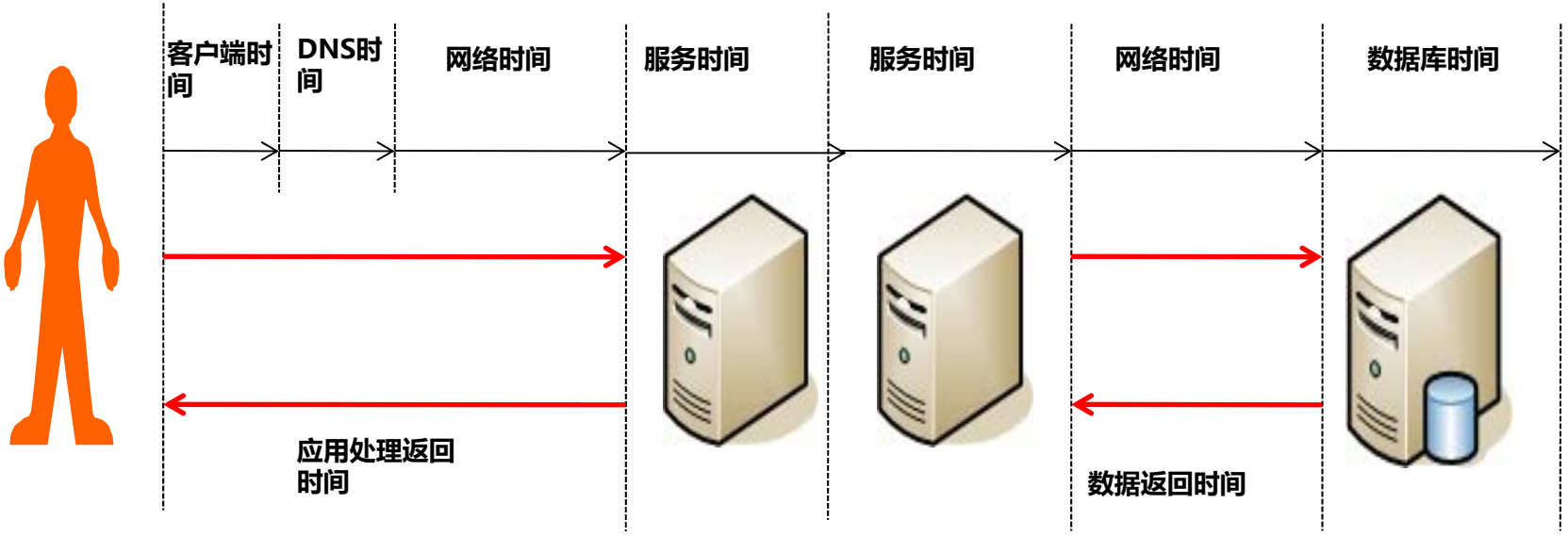


测试执行和监控

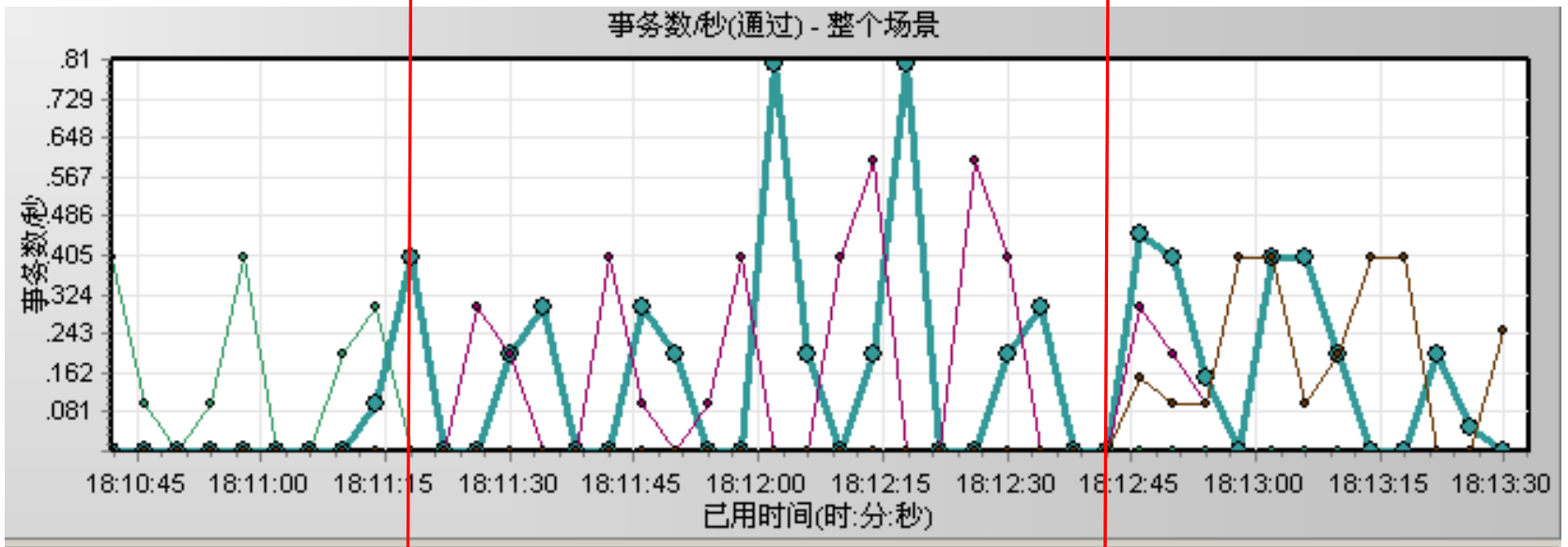
- 性能监控：

① Web/应用/数据库服务器

②系统资源、应用资源、硬件资源

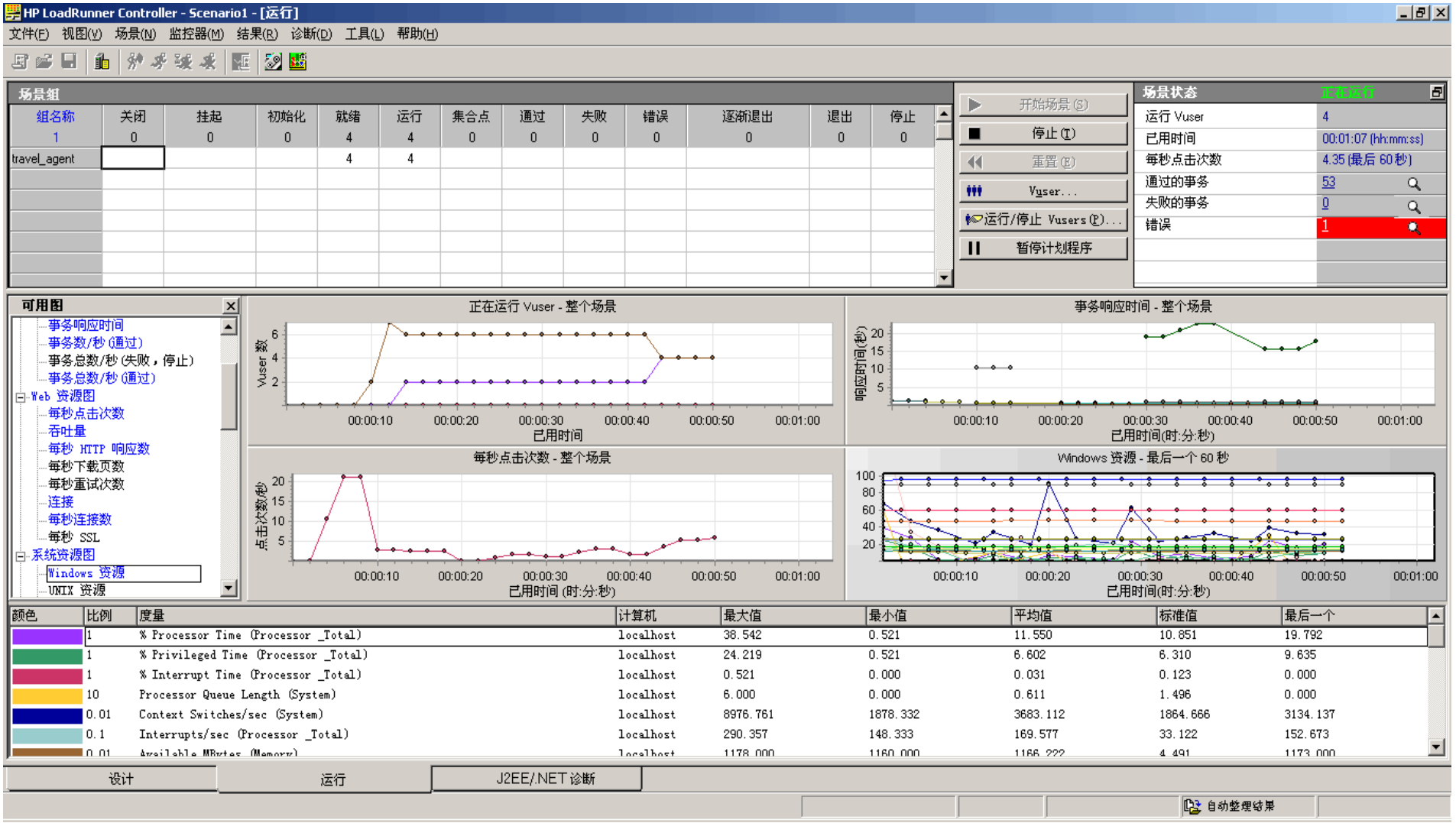


示例：监控



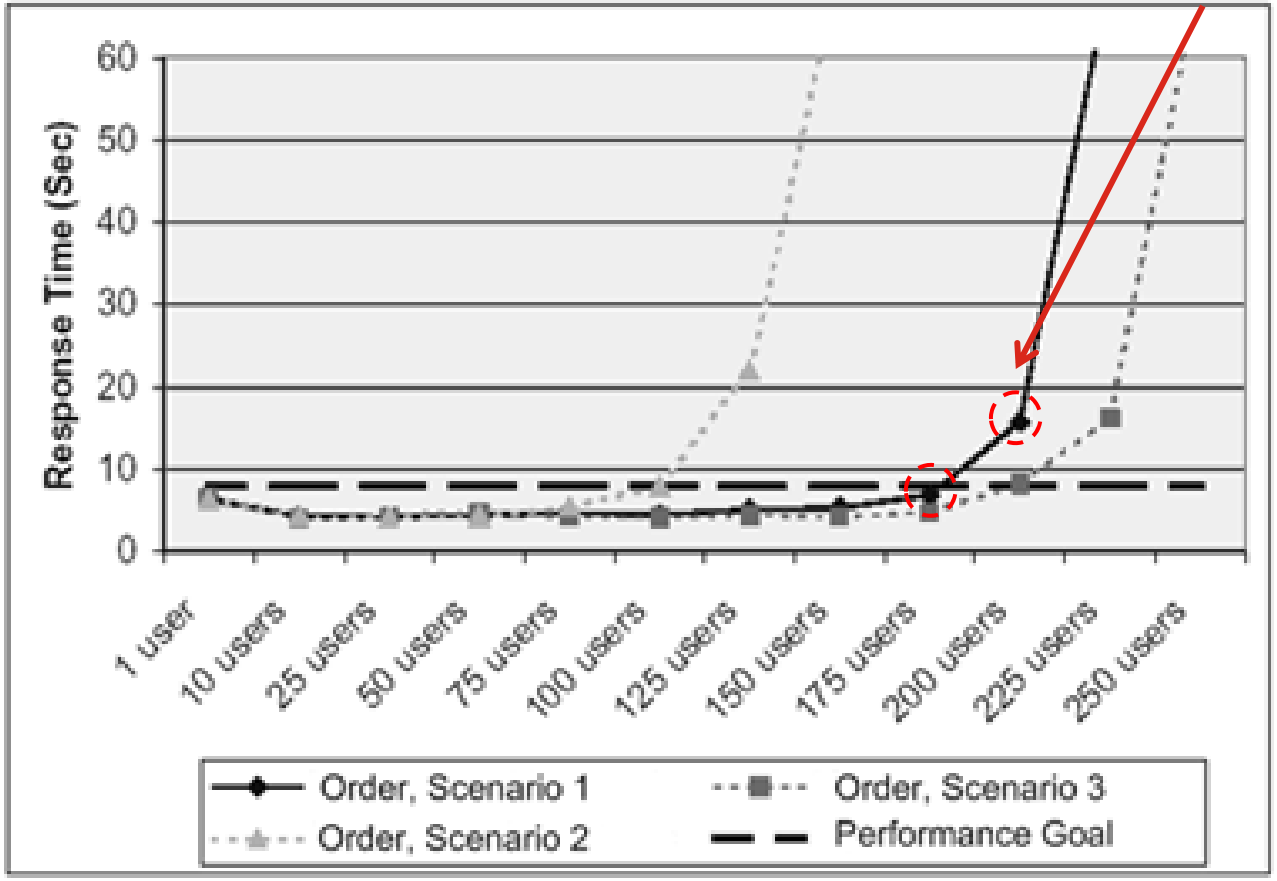
颜色	比例	事务	最大值	最小值	平均值	标准值	最后一个
	1	vuser_init_Transaction	暂缺	暂缺	0.056	暂缺	0.000
	1	Book	暂缺	暂缺	0.145	暂缺	0.000
	1	Action_Transaction	暂缺	暂缺	0.145	暂缺	0.250
	1	vuser_end_Transaction	暂缺	暂缺	0.058	暂缺	0.250

示例：结果分析


HP LoadRunner Controller - Scenario1 - [运行]
文件(F) 视图(V) 场景(N) 监控器(M) 结果(R) 诊断(D) 工具(L) 帮助(H)
场景组
组名称 关闭 挂起 初始化 就绪 运行 集合点 通过 失败 错误 逐渐退出 退出 停止
travel_agent
开始场景(S)
停止(T)
重置(E)
Vuser...
运行/停止 Vusers(P)...
暂停计划程序
场景状态
运行 Vuser
已用时间
每秒点击次数
通过的事务
失败的事务
错误
可用图
事务响应时间
事务数/秒(通过)
事务总数/秒(失败，停止)
事务总数/秒(通过)
Web 资源图
每秒点击次数
吞吐量
每秒 HTTP 响应数
每秒下载页数
每秒重试次数
连接
每秒连接数
每秒 SSL
系统资源图
Windows 资源
UNIX 资源
正在运行 Vuser - 整个场景
事务响应时间 - 整个场景
每秒点击次数 - 整个场景
Windows 资源 - 最后一个 60 秒
颜色 比例 度量 计算机 最大值 最小值 平均值 标准值 最后一个
1 % Processor Time (Processor _Total) localhost 38.542 0.521 11.550 10.851 19.792
1 % Privileged Time (Processor _Total) localhost 24.219 0.521 6.602 6.310 9.635
1 % Interrupt Time (Processor _Total) localhost 0.521 0.000 0.031 0.123 0.000
10 Processor Queue Length (System) localhost 6.000 0.000 0.611 1.496 0.000
0.01 Context Switches/sec (System) localhost 8976.761 1878.332 3683.112 1864.666 3134.137
0.1 Interrupts/sec (Processor _Total) localhost 290.357 148.333 169.577 33.122 152.673
设计
运行
J2EE/.NET 诊断
自动整理结果

结果分析

- 捕捉被监控的数据曲线发生突变的地方——拐点

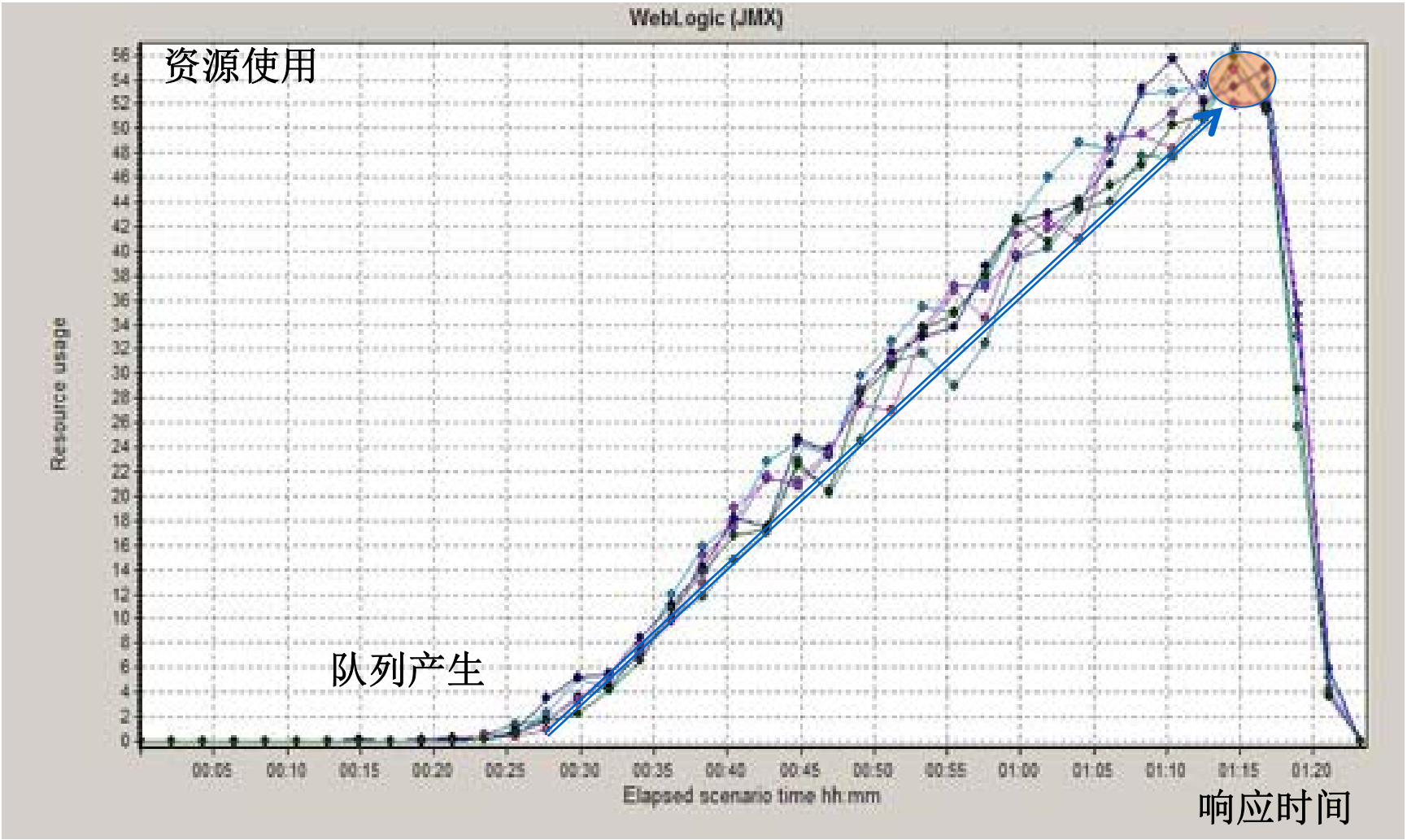


响应时间分析

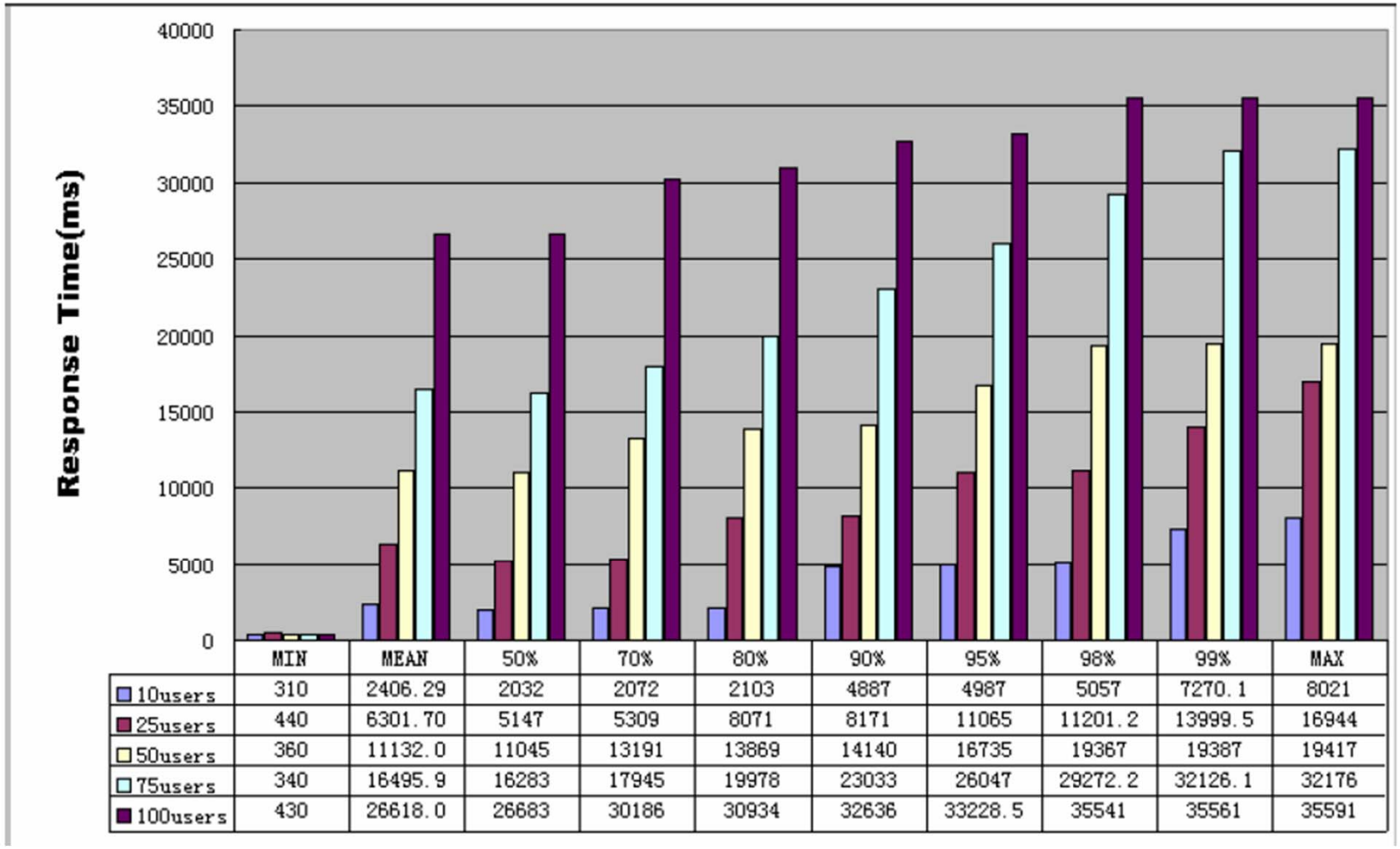
- 在某一点上，执行队列开始增长，因为服务器上所有的线程都已投入使用，传入的请求不再被立即处理，而是放入队列中，当线程空闲时再处理。

- 当系统达到饱和点，服务器吞吐量保持稳定后，就达到了给定条件下的系统上限。但是，随着服务器负载的继续增长，响应时间也随之延长，虽然吞吐量保持稳定。

响应时间分析

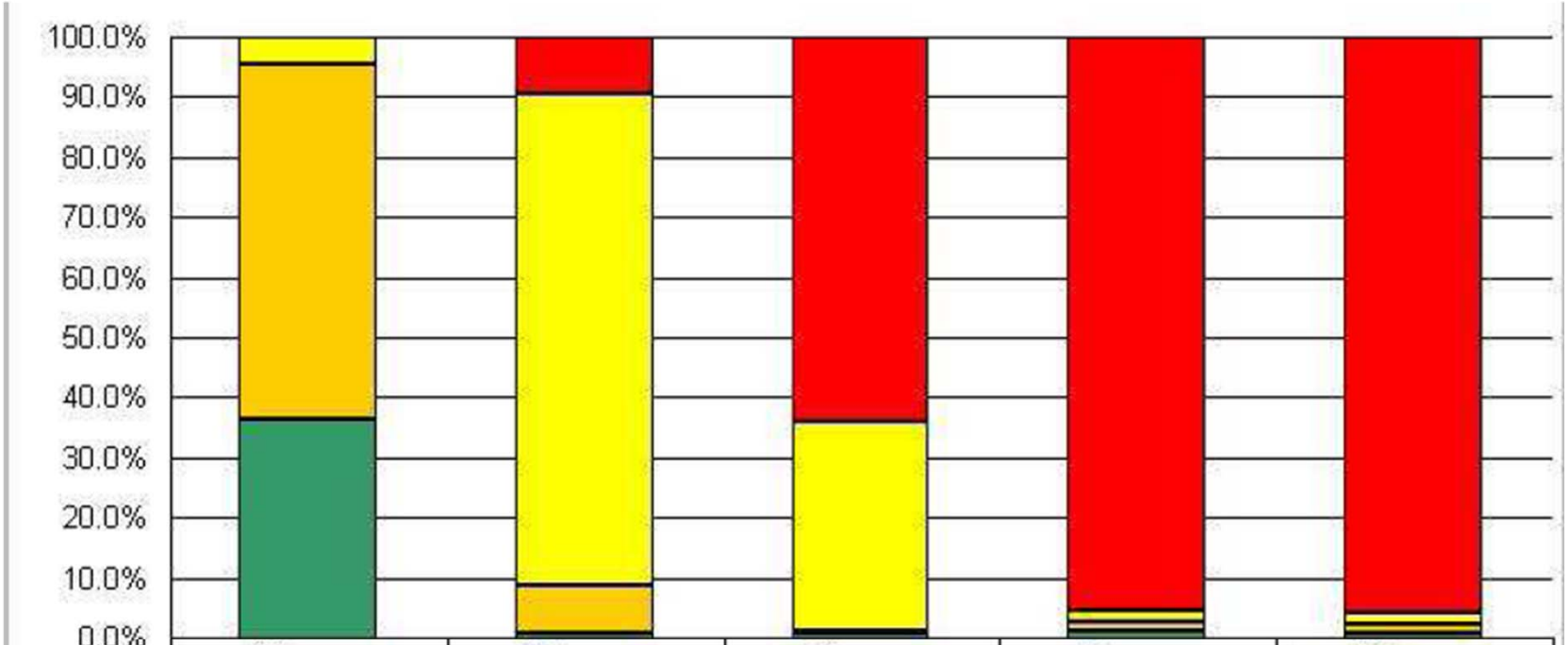

WebLogic (JMX)
资源使用
Resource usage
队列产生
Elapsed scenario time hh:mm
响应时间

响应时间分析



	MIN	MEAN	50%	70%	80%	90%	95%	98%	99%	MAX
10users	310	2406.29	2032	2072	2103	4887	4987	5057	7270.1	8021
25users	440	6301.70	5147	5309	8071	8171	11065	11201.2	13999.5	16944
50users	360	11132.0	11045	13191	13869	14140	16735	19367	19387	19417
75users	340	16495.9	16283	17945	19978	23033	26047	29272.2	32126.1	32176
100users	430	26618.0	26683	30186	30934	32636	33228.5	35541	35561	35591

响应时间分析



	10users	25users	50users	75users	100users
50000	0.00%	9.60%	64.10%	95.60%	95.80%
10000	4.70%	81.80%	34.80%	1.60%	1.80%
5000	58.80%	7.70%	0.20%	1.50%	1.50%
2000	36.50%	0.90%	0.90%	1.30%	0.90%

Response Time(ms)

分析请求失败的原因

- 请求超时
- 连接被拒绝
- 应用服务器error
- 数据库error
- 系统崩溃
- …

性能测试要点 -1

- 测试环境应尽量与产品运行环境保持一致，应单独运行尽量避免与其他软件同时使用
- 录制脚本和手工编写脚本相结合
- 重点在于前期数据的设计与后期数据的分析
- 设置数据池，实现变量加载
- 需要同时监控数据库服务器、Web服务器以及网络资源等使用情况，以便对系统的性能做全面评估

性能测试要点 -2

- 模拟用户数的递增
- 合理设置交易之间时间间隔
- 超时（timeout）的设置、错误处理
- 并发用户连续执行交易数的设置
- 并发用户数量极限点
- 尽量将执行负载测试的机器合理分布
- 加压机器的CPU使用率也有必要监控

性能测试工具

- 能模拟实际用户的操作行为，记录和回放多用户测试中的事务处理过程，自动生成相应的测试脚本
- 能针对脚本进行修改，增加逻辑控制、完成参数化和数据关联
- 可以设置不同的应用环境和场景，通过虚拟用户执行相应的测试脚本
- 通过系统监控工具获得系统性能的相关指标的值

开源工具

- **Apache Flood** （http://httpd.apache.org/test/flood/）一个Web性能测试工具
- **Gatling**(http://gatling-tool.org)是一款基于Scala 开发的Web高性能服务器性能测试工具
- **nGrinder**（http://www.nhnopensource.org/ngrinder/），Web 分布式压力测试系统
- **Siege**（http://www.joedog.org/JoeDog/Siege）是一个开源的Web压力测试和评测工具。
- **DBMonster**是一个生成随机数据、用来测试SQL数据库的压力测试工具，详见http://dbmonster.kernelpanic.pl/。
- 更多的性能测试工具，可访问 http://www.opensourcetesting.org/performance.php

商业工具

- HP LoadRunner
- IBM Rational Performance Tester
- Radview WebLoad
- Compuware QA Load
- Quest Benchmark Factory
- 微软WAS（Web Access Stress test）
- Paessler Webserver Stress Tool
- MINQ PureLoad

内容提要

- 理解“性能”与“性能测试”
- 压力测试
- 性能测试的过程与实践